Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 155 (2007) 365–373

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

# O(1) query time algorithm for all pairs shortest distances on permutation graphs

Alan P. Sprague

*Department of Computer and Information Sciences, University of Alabama at Birmingham, Birmingham, AL 35294-1170, USA*

Received 17 December 2005; received in revised form 4 June 2006; accepted 29 June 2006
Available online 22 August 2006

**Abstract**

We present an algorithm for the all pairs shortest distance problem on permutation graphs. Given a permutation model for the graph on $n$ vertices, after $O(n)$ preprocessing the algorithm will deliver answers to distance queries in $O(1)$ time. In the EREW PRAM model, preprocessing can be accomplished in $O(\log n)$ time with $O(n)$ work. Where the distance between query vertices is $k$, a path can be delivered in $O(k)$ time. The method is based on reduction to bipartite permutation graphs, a further reduction to unit interval graphs, and a coordinatization of unit interval graphs.


*Keywords:* All pairs shortest paths; Bipartite permutation graph; Permutation graph; Unit interval graph

## 1. Introduction

Let $\pi$ be a permutation of $\{1, 2, \ldots, n\}$. The graph $G(\pi)$ is defined as the graph having $\{1, 2, \ldots, n\}$ as vertex set, in which vertices $i$ and $j$ are adjacent iff $(i - j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0$. A graph $G$ on $n$ vertices is called a *permutation graph* if $G$ is isomorphic to $G(\pi)$ for some permutation $\pi$ of $\{1, 2, \ldots, n\}$.

A *repetitive mode* algorithm [15] for a problem consists of a preprocessing algorithm and a query handling algorithm. For the all pairs shortest distance (APSD) problem, a query takes the form: find $\text{dist}_G(v, w)$, where vertices $v$ and $w$ are specified in the query. Here $\text{dist}_G(v, w)$, the distance in graph $G$ from $v$ to $w$, is the number of edges in a shortest path from $v$ to $w$.

The classic algorithm for the all pairs shortest path (APSP) problem in graphs is the $O(n^3)$ algorithm of Floyd [6]. Substantial effort has been given to developing $o(n^3)$ algorithms for APSP on graphs in general; a couple of the most recent are [1,9,10]. Much work has also gone into algorithms for APSP or APSD on special classes of graphs. Interval graphs and circular arc graphs have attracted the most attention. Initially $O(n^2)$ algorithms were developed [14,16], and then repetitive mode algorithms with $O(n)$ preprocessing time and $O(1)$ query time [3,12,19]; optimal parallel algorithms with $O(\log n)$ execution time (EREW or CREW) were also given in the same three papers. For other classes, $O(n^2)$ algorithms have been given for bipartite permutation graphs (bpg's) [4], strongly chordal graphs [8], and chordal bipartite graphs [11].

*E-mail address:* sprague@uab.edu.

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2006.06.008

In this paper we develop an algorithm that, on a permutation graph of $n$ vertices, can deliver a reply to a distance query in O(1) time after an initial O($n$) preprocessing time. Thus for the purpose of computing all $n^2$ distances, this algorithm takes O($n^2$) time.

Since the algorithm can deliver the first internal vertex in a shortest path as well as the distance, the entire path can be delivered in O($k$) time where $k$ is the distance between the query points $s$ and $t$. (This can be accomplished by internally generating $k-2$ queries for the first internal vertex in the path from the previously returned vertex to $t$).

As noted in [3], O($n$) preprocessing time repetitive mode algorithms require only O($n$) space, while the standard approach to APSD and APSP uses O($n^2$) space.

A *bipartite permutation graph* (bpg) is defined as its name indicates: a permutation graph that is bipartite. An *interval graph* is a graph whose vertices may be put into one-to-one correspondence with a set of closed intervals on the number line, such that two vertices are adjacent iff the corresponding intervals are nondisjoint; this set of intervals is called a *model* for the graph. A *unit interval graph* (uig) is an interval graph that has a model in which all intervals have length 1.

The method we use to solve the APSD problem emphasizes reductions. In Section 2 we reduce APSD on permutation graphs to the same problem on bpg's, and then in Section 4 we perform a reduction from bpg's to unit interval graphs, and (in the same section) we solve APSD on unit interval graphs. The fairly short Section 5 draws together the pieces of the preprocessing algorithm from Sections 2 and 4, gives pseudocode for the query handling algorithm developed in the same two sections, describes the parallel algorithm, and displays an example.

In Section 3 we establish a characterization for bpg's; by changing a single word, we obtain a characterization of unit interval graphs. These characterizations, although implicit in [5,18], were not previously stated, and are needed for the reduction in Section 4.

We will presume that the graph is connected; it is not hard to extend this to permutation graphs that are not connected.

## 2. Reduction to bpg's

In this section we first introduce permutation models (also called permutation diagrams), and then reduce the APSD problem on permutation graphs to APSD on bpg's. The permutation model will be our key conceptual tool for permutation graphs.

Let $V$ be a set of $n$ line segments, to be called interchangeably *vertices* or *line segments*. Each line segment has one end (called its *top pin*) on a horizontal line $h_1$, and has its other end (called its *bottom pin*) on a horizontal line $h_2$ below $h_1$. Top pins are labeled $1, 2, \ldots, n$ from left to right (no two coincide), and bottom pins are likewise labeled. We define two vertices to be *adjacent* if they, as line segments, intersect. This set of line segments is called a *permutation model* for the resulting graph. It is known that a graph is a permutation graph iff it has a permutation model [7].

Let $G$ be a permutation graph on $n$ vertices, with permutation model defined by a set $V$ of $n$ line segments. For each vertex $w$, we denote the position of the top pin of $w$ by $w_T$. The function v2t gives this bijection from vertices to $\{1, 2, \ldots, n\}$: $\text{v2t}(w) = w_T$. Likewise, $w_B$ is the position of the bottom pin of $w$, and $\text{v2b}(w) = w_B$. For example, in Fig. 1, $a_T = 1$, $b_T = 2$, $a_B = 4$, $b_B = 1$.

Given a permutation $\pi$ of $\{1, 2, \ldots, n\}$, a model for $G(\pi)$ may be specified by setting $\text{v2t}(i) = i$ and $\text{v2b}(i) = \pi^{-1}(i)$ for all $i$.

Let $v$ and $w$ be adjacent vertices. If $v_T < w_T$ (hence $v_B > w_B$) then we say that the directed edge $vw$ is a *clockwise turn*, and directed edge $wv$ is a *counterclockwise turn*. A vertex $v$ is said to be *clockwise maximal* (also written as *cw-maximal*) if for every vertex $w$ adjacent to $v$, $wv$ is a clockwise turn. Also, $v$ is *ccw-maximal* if for every vertex $w$ adjacent to $v$, $wv$ is a counterclockwise turn. Let $V_{cw}$ be the set of cw-maximal vertices and $V_{ccw}$ be the set of

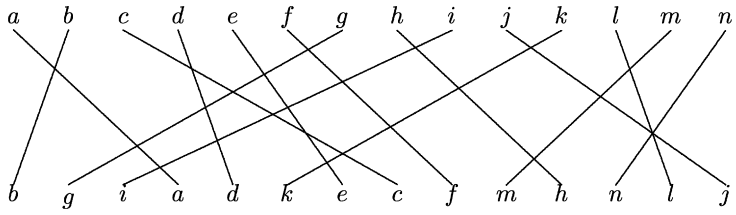


Fig. 1. Permutation model for permutation graph $G$.

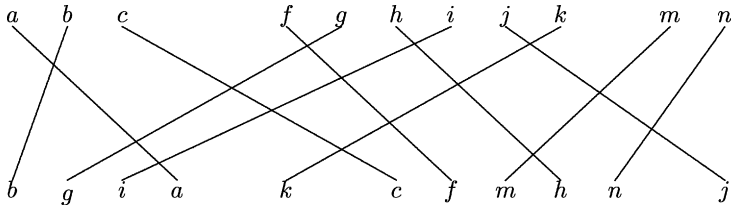


Fig. 2. Permutation model for bpg $H$ (before closing up gaps). Vertices of $H$ are the cw-maximal and ccw-maximal vertices from Fig. 1.

ccw-maximal vertices. In Fig. 1, $dc$ is a counterclockwise turn and $dg$ is clockwise. Vertex $c$ is ccw-maximal, $g$ is cw-maximal, and $d$ and $e$ are neither.

$V_{ccw}$ is an anticlique, since each edge $vw$ provides evidence that one of its two vertices is not ccw-maximal. (As usual, an *anticlique* is a set of vertices such that no two are joined by an edge, while a *clique* is a set of vertices, each of which is a neighbor of every other.) Likewise, $V_{cw}$ is an anticlique.

We next define two arrays to link together the top pins and bottom pins of the same vertex: for each vertex $v$, $\text{t2b}[v_T] = v_B$ and $\text{b2t}[v_B] = v_T$. These arrays may be computed in $O(n)$ time, given v2t and v2b. In Fig. 1, $\text{t2b}[1] = 4$, since $\text{v2t}(a) = 1$ and $\text{v2b}(a) = 4$.

Define the array prefix_max_t2b by $\text{prefix\_max\_t2b}[i] = \max\{\text{t2b}[j] : j \leqslant i\}$. It is not hard to see that for a vertex $v$, $v$ is in $V_{ccw}$ iff there is no vertex $w$ with $w_T < v_T$ and $w_B > v_B$ iff $\text{prefix\_max\_t2b}[v_T] = v_B$. Since array prefix_max_t2b may be computed in $O(n)$ time (given t2b), this gives us an $O(n)$ method to construct $V_{ccw}$. Similarly, $V_{cw}$ may be constructed in $O(n)$ time.

This same prefix maxima process also yields a second benefit: for each vertex $v$ that is not ccw-maximal, $\text{prefix\_max\_t2b}[v_T]$ points to the rightmost member of $V_{ccw}$ intersecting $v$, as is established in the next proposition.

**Proposition 2.1.** *Let $v \notin V_{ccw}$ and $\text{prefix\_max\_t2b}[v_T] = w_B$. Then $w$ is the rightmost member of $V_{ccw}$ that intersects $v$.*

**Proof.** $\text{prefix\_max\_t2b}(v_T) = w_B$ where $w$ has the rightmost bottom pin, among those segments whose top pin is not right of $v_T$. Since prefix_max_t2b only points to vertices in $V_{ccw}$, $w \in V_{ccw}$. Since $w_T < v_T$ and $w_B > v_B$ (since $w \neq v$ and $w_B = \max\{x_B : x_T \leqslant v_T\}$), $w$ intersects $v$. Then, among vertices in $V_{ccw}$ that intersect $v$, $w$ is the one with rightmost bottom pin. □

For each $v \notin V_{ccw}$, define $R_{ccw}(v)$ to be the rightmost member of $V_{ccw}$ that intersects $v$. By Proposition 2.1, this function may be computed in $O(n)$ time by $R_{ccw}(v) = \text{b2v}(\text{prefix\_max\_t2b}[v_T])$. Similarly, for $v \notin V_{cw}$, define $R_{cw}(v) = \text{t2v}(\text{prefix\_max\_b2t}[v_B])$. Finally, for $v \notin V_{ccw}$ (respectively, $v \notin V_{cw}$) define $L_{ccw}(v)$ (resp., $L_{cw}(v)$) to be the leftmost member of $V_{ccw}$ (resp. $V_{cw}$) that intersects $v$; these functions may be computed by taking suffix minima.

Let $H$ be the induced subgraph of $G$ on $V_{cw} \cup V_{ccw}$. $H$ is bipartite (since $V_{cw}$ and $V_{ccw}$ are anticliques). $H$ is a permutation graph: a permutation model for $H$ may be constructed by deleting vertices not in $H$ from the permutation model for $G$ (and closing up gaps). For the permutation graph in Fig. 1, the induced bpg on cw-maximal and ccw-maximal vertices is displayed in Fig. 2.

The upcoming lemma plays a role in the proof of the subsequent proposition. (The lemma is a reflection of the fact that permutation graphs are comparability graphs, where clockwise turn is used as the method of directing edges).

**Lemma 2.1.** *If $v_1v_2$ and $v_2v_3$ are both clockwise turns then $v_1$ intersects $v_3$.*

**Proof.** The hypotheses imply that $v_{1,T} < v_{2,T} < v_{3,T}$ and $v_{1,B} > v_{2,B} > v_{3,B}$. □

The following proposition shows that when seeking a shortest path between two vertices $s$ and $t$ in $G$, one may restrict attention to paths, all of whose intermediate vertices are in $H$.

**Proposition 2.2.** *For all vertices $s, t \in G$, $\text{dist}_G(s,t) = \text{dist}_{H'}(s,t)$ where $H' = H \cup \{s,t\}$.*

**Proof.** The conclusion is trivial if $\text{dist}_G(s, t) \leqslant 1$, so we assume that $s$ and $t$ are nonintersecting segments. Without loss of generality we assume that $s$ is left of $t$.

Let $p = (v_0, v_1, \ldots, v_r)$ (where $v_0 = s$ and $v_r = t$) be a shortest path from $s$ to $t$ in $G$. Then $p$ is chordless. (In any graph, a path with a chord can be shortened.) Since $p$ is chordless, by Lemma 2.1 $p$ does not contain two consecutive clockwise turns; similarly, $p$ does not contain two consecutive counterclockwise turns.

We further assume that, among all shortest paths from $s$ to $t$, $p$ contains the fewest vertices not in $H$.

Suppose that $p$ contains a vertex $v_i$ $(1 \leqslant i \leqslant r-1)$ that is not in $H$, i.e. a segment that is neither cw-maximal nor ccw-maximal. We may suppose that the directed edge $v_{i-1}v_i$ is a counterclockwise turn (and hence $v_i v_{i+1}$ is clockwise). Define $w$ such that $w_B = \text{prefix\_max\_t2b}[v_{i,T}]$. By Proposition 2.1, $w \in C_{ccw}$ and $w$ intersects $v_i$. Then $w$ intersects both $v_{i-1}$ and $v_{i+1}$, by Lemma 2.1. Define $q$ to be the sequence of segments obtained from $p$ by replacing $v_i$ by $w$. Then $q$ is a path of the same length as $p$, and the number of vertices of $q$ outside of $H$ is one less than the corresponding number for $p$. This contradicts the supposed minimality of $p$. Hence every vertex in $p$ (except perhaps $s$ and $t$) is in $H$. □

**Proposition 2.3.** *For nonadjacent vertices $s, t \in G$, let $H' = H \cup \{s, t\}$. We presume that as line segments $s$ is left of $t$ (in the contrary case, exchange $s$ and $t$).*

(a) *If* $\text{dist}_{H'}(s, t) \geqslant 3$, $\text{dist}_G(s, t) = 2 + \min\{\text{dist}_H(R_i(s), L_j(t) : i, j \in \{cw, ccw\}\}$.
(b) *If* $\text{dist}_{H'}(s, t) = 2$ *then* $(s, R_{cw}(s), t)$ *or* $(s, R_{ccw}(s), t)$ *is a path.*

**Proof.** Let $\text{dist}_G(s, t) \geqslant 2$. By Proposition 2.2, there is a shortest path $s = v_0, v_1, \ldots, v_r = t$ where each $v_i$ $(1 \leqslant i \leqslant r-1)$ is in $H$; call this path, $p$.

We presume that $v_1$ is cw-maximal. (The alternate case, where $v_1$ is ccw-maximal, is handled similarly.) Let $w = R_{cw}(s)$. We assume that $w \neq v_1$, and show that $w$ may be substituted for $v_1$ in $p$.

We show that $w$ intersects $v_2$. $v_{2,T} < w_T$, because $v_{2,T} < v_{1,T} < w_T$. $v_{2,B} > w_B$, because $w_B < s_B < v_{2,B}$ ($s_B < v_{2,B}$ because $p$, a shortest path to $t$ (which is right of $s$), must be chordless).

Hence replacing $v_1$ by $w$ in path $p$, we obtain a new path, to be called $q$, from $s$ to $t$. This completes the proof of (b). To complete the proof of (a), apply the same process on path $q^{-1}$ from $t$ to $s$, to replace $v_{r-1}$ by $L_{cw}(t)$ or $L_{ccw}(t)$). □

Note that if $s \in V_{cw}$ then $R_{cw}(s)$ may be considered to be vacant: the vertex following $s$ in a path from $s$ will not be another vertex of $V_{cw}$. Hence in this case, 2 of the 4 distances in (a) may be taken as $\infty$ (and if $t$ is also in $V_{cw} \cup W_{ccw}$, 3 of the 4 are $\infty$).

**Proposition 2.4.** *If there is an algorithm for APSD on bpg's on n vertices using* O($n$) *preprocessing time and* O(1) *query time, then there is an algorithm for APSD on permutation graphs on n vertices using* O($n$) *preprocessing time and* O(1) *query time.*

**Proof.** Let $G$ be a permutation graph on $n$ vertices, specified by a permutation model. Let $H$ be the subgraph of $G$ induced by the cw-maximal and ccw-maximal vertices of $G$.

Given the model for $G$, in O($n$) time we can compute array prefix_max_t2b: from this we can determine which vertices are ccw-maximal and compute function $R_{ccw}$. Similarly, in O($n$) time we can determine which vertices are cw-maximal and hence construct $H$. Finally, functions $R_{cw}$, $L_{ccw}$, and $L_{cw}$ can be computed like $R_{ccw}$.

Given a query requesting the distance between two vertices $s$ and $t$ of $G$, in O(1) time we can determine if $s$, as a line segment in the model, intersects $t$. If yes, then the distance is 1, and if not, the distance can be computed in O(1) time by using Proposition 2.3. □

## 3. bpg's and unit interval graphs

In this section we develop characterizations for bpg's and uig's. Some of the results of this section were proved in [5,18]. Where appropriate, we refer to the proofs in those papers.

Let $G$ be a connected graph (not necessarily a uig or bpg), and $z \in V(G)$. For each nonnegative integer $i$ define $L_i$ to be the set of vertices at distance $i$ from $z$. This partitions $V(G)$ into sets $L_i$ (and $L_0 = \{z\}$). This partition plays a well-known role in performing a breadth first search (BFS) from $z$; we call this partition, BFS($z$). For a vertex $x$ in $L_i$, PrevN($x$) represents the set of neighbors of $x$ in $L_{i-1}$ and NextN($x$) the neighbors in $L_{i+1}$.

We say that BFS($z$) has the *coherence property* if for each $i$ and vertices $x$ and $y$ in $L_i$,

(a) PrevN($x$) $\subseteq$ PrevN($y$) or PrevN($x$) $\supseteq$ PrevN($y$).
(b) NextN($x$) $\subseteq$ NextN($y$) or NextN($x$) $\supseteq$ NextN($y$).
(c) If PrevN($x$) $\subset$ PrevN($y$), then NextN($x$) $\supseteq$ NextN($y$).

In light of (a) and (b), (c) is equivalent to denying that PrevN($x$) and NextN($x$) simultaneously are proper subsets of PrevN($y$) and NextN($y$) (respectively); hence in effect PrevN() and NextN() play symmetric roles in (c).

The next lemma talks about an ordering of $V(G)$; by an *ordering* of $V(G)$ we mean a bijection between $V(G)$ and $\{1, 2, \ldots, |V(G)|\}$.

**Lemma 3.1.** *Let $z \in V(G)$. Let* BFS($z$) *satisfy the coherence property. Define an order on vertices based*

(i) *on their levels in the* BFS, *and*
(ii) *among the vertices in the same level, on increasing order of $|NextN(x)| - |PrevN(x)|$, breaking ties arbitrarily.*

*Then for each vertex $x$ (say $x \in L_i$), $PrevN(x) \cup L_i \cup NextN(x)$ is a set of consecutive vertices.*

**Proof.** Let $x \in L_i$, $v \in L_{i+1}$, and $x$ be adjacent to $v$. Let $u$ be in $L_{i+1}$ and $u$ precede $v$ in the ordering; then $|\text{NextN}(u)| - |\text{PrevN}(u)| \leqslant |\text{NextN}(v)| - |\text{PrevN}(v)|$. It is not difficult to see that the coherence property implies that PrevN($u$) $\supseteq$ PrevN($v$); then since $v$ is adjacent to $x$, $u$ is also adjacent to $x$. Summarizing, we have shown that if $x$ is adjacent to $v$, then $x$ is adjacent to all vertices in $L_{i+1}$ preceding $v$. Similarly, if $x$ is adjacent to a vertex $w$ in $L_{i-1}$, then $x$ is adjacent to all vertices in $L_{i-1}$ following $w$. The conclusion follows. □

**Corollary 3.2.** *If in addition each $L_i$ is a clique then $G$ is a uig.*

**Proof.** This ordering of the vertices has the property that for each $x$, the closed neighborhood of $x$ is a set of consecutive vertices; Roberts [17] showed that this implies the graph is a uig. □

**Corollary 3.3.** *If in addition each $L_i$ is an anticlique then $G$ is a bpg.*

**Proof.** If each $L_i$ is an anticlique, then $G$ is bipartite: the two sides of the bipartition are the union of the even levels, and the union of the odd levels. Proposition 3.2 of [18] completes the proof. □

We conclude with characterizations of uig's and bpg's. For uig's, one direction is proved above in Corollary 3.2 and the other is proved as Proposition 2.1 of [5]. For bpg's, one direction is proved above in Corollary 3.3 and the other is proved as Observations 3.1 and 3.3 of [18]. This characterization of bpg's may also be found in [2].

**Proposition 3.1.** *A connected graph $G$ is a uig iff there is a vertex $z$ of $G$ so that the partition* BFS($z$) *satisfies the coherence property, and each level of* BFS($z$) *is a clique.*

**Proposition 3.2.** *A connected graph $G$ is a bpg iff there is a vertex $z$ of $G$ so that the partition* BFS($z$) *satisfies the coherence property, and each level of* BFS($z$) *is an anticlique.*

In both these propositions a vertex may play the role of $z$ iff it is an extreme vertex in some model; for uig's, this means that there is a model so that the vertex is the leftmost interval in the model; for bpg's, this means that there is a model so that the vertex has leftmost top pin in the model. Both propositions have an immediate extension to graphs that are not connected: apply the proposition to each component separately.

## 4. Reduction to unit interval graphs

In this section we describe the reduction from the APSD problem on bpg's to the same problem on uig's, and the solution of the problem on uig's. We are given a permutation model for bpg $H$ on $n$ vertices. Let the line segment with leftmost top pin be called $a$.

For $i \geqslant 0$, let $L_i$ be the set of vertices of $H$ at distance $i$ from $a$. BFS($a$) satisfies the coherence property, by Proposition 3.2. Define $X$ to be the set of all unordered pairs of vertices $v$, $w$ such that $v$ and $w$ are in the same level.

Define $H^*$ to be the following graph. $V(H^*) = V(H)$, and $E(H^*) = E(H) \cup X$; then $H^*$ differs from $H$ in that in $H^*$ each $L_i$ is a clique instead of an anticlique. In $H^*$, the set of vertices at distance $i$ from $a$ is precisely $L_i$ (since adding edges between vertices at the same distance from $a$ does not change distance from $a$). By Proposition 3.1, $H^*$ is a uig and $a, L_1, L_2, \ldots$ is a BFS from leftmost interval $a$ in some interval model.

**Lemma 4.1.** *Let vertices $u$ and $v$ be in level $L_i$ $(i \geqslant 1)$. Then $u$ and $v$ have a common neighbor in level $L_{i-1}$.*

**Proof.** PrevN($v$) and PrevN($u$) are nonempty. By the coherence property, one is a subset of the other. Then every vertex of the smaller one is a vertex of $L_{i-1}$ that is adjacent to both. □

Define the function *round*, from $V(H) \times V(H) \times \mathbb{Z}$ to $\mathbb{Z}$, as follows. round$(v, v', j) = k$ where $k = j$ or $k = j + 1$, and $k$ is even iff $v$ and $v'$ are both in $V_{cw}$ or are both in $V_{ccw}$. Hence round$(v, v', j)$ may be thought of as rounding $j$ up to an integer having the same parity as $\text{dist}_H(v, v')$. Since $H$ is a bipartite graph, the parity of $\text{dist}_H(v, v')$ is known even when the distance is not.

The next proposition says that to compute distances in $H$, it is sufficient to compute distances in $H^*$ and round up to the appropriate parity.

**Proposition 4.1.** *Let $x$ and $y$ be vertices, with $x \in L_i$, $y \in L_j$, and $i \leqslant j$. Then*:

(a) $\text{dist}_{H^*}(x, y)$ *equals either* $j - i$ *or* $j - i + 1$.
(b) $\text{dist}_H(x, y) = \textit{round}(x, y, \text{dist}_{H^*}(x, y))$.

**Proof.** (a) (Although this is clear in [19], for completeness, we prove it here.) In any BFS structure, the distance between two vertices is at least the change in level from one to the other; hence $\text{dist}_{H^*}(x, y) \geqslant j - i$. A path of length $j - i + 1$ (at most) between $x$ and $y$ may be constructed by (i) constructing a path of length $j - i$ (i.e., $j - i$ edges) by traveling from $y$ toward $a$, until arriving at a vertex (say $u$) of $L_i$; then (ii) if $u \neq x$, appending edge $(u, x)$ to the path ($(u, x)$ is an edge since $L_i$ is a clique in $H^*$).

(b) We will call an edge of $H^*$ having both vertices in the same level, an *intralevel* edge. In the proof of (a) we saw that either (i) $\text{dist}_{H^*}(x, y) = j - i$ and there is a path $p$ of length $j - i$ from $y$ to $x$ having no intralevel edges, or (ii) $\text{dist}_{H^*}(x, y) = j - i + 1$ and there is a path $q$ of length $j - i + 1$ from $y$ to $x$ having exactly one intralevel edge. In case (i), $p$ is also a path in $H$, so $\text{dist}_H(x, y) = \text{dist}_{H^*}(x, y)$. Also conversely, if $\text{dist}_H(x, y) = j - i$, then case (i) holds, i.e. $\text{dist}_{H^*}(x, y) = j - i$, since $\text{dist}_H(x, y) \geqslant \text{dist}_{H^*}(x, y)$ (from $E(H) \subseteq E(H^*)$). In case (ii), $q$ has exactly one intralevel edge. Let $e$ be the intralevel edge in $q$. By Lemma 4.1, there is a path of length 2 in $H$ joining the two ends of $e$, so there is a path of length $j - i + 2$ in $H$ from $y$ to $x$; also, $H$ has no path of length $j - i + 1$ because $H$ is bipartite.

Summarizing, either $\text{dist}_H(x, y) = \text{dist}_{H^*}(x, y) = j - i$ or $\text{dist}_H(x, y) = \text{dist}_{H^*}(x, y) + 1 = j - i + 2$; this may be written as $\text{dist}_H(x, y) = \text{round}(x, y, \text{dist}_{H^*}(x, y))$ since $x$ and $y$ are both cw-maximal or both ccw-maximal iff $j - i$ is even. □

On the vertices of bpg $H$ we define an ordering $\sigma$; $\sigma$ orders vertices according to their level in BFS($a$) and, within each level, from left to right. Then $\sigma$ satisfies conditions (i) and (ii) stated in Lemma 3.1.

In order to use tree traversal concepts on BFS($a$) we define the notion of parent: for each vertex $v$ ($v \neq a$) we say that vertex $w$ is a *parent* of $v$ if $w$ is the first vertex (according to $\sigma$) that is a neighbor of $v$. This defines a tree, with $a$ as root. In bpg $H$, $w$ is the parent of $v$ iff (where $v \in V_{cw}$) $w = L_{ccw}(v)$ or (where $v \in V_{ccw}$) $w = L_{cw}(v)$. For example, in Fig. 3, $\sigma$ orders vertices by column, and, within column, from top to bottom.

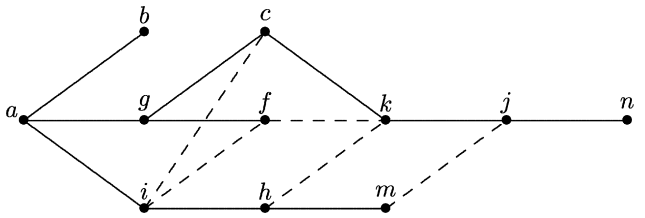


Fig. 3. Breadth first search (from extremal vertex $a$) of the bpg $H$ of Fig. 2. Parent–child edges $wv$ where $w = L_{cw}(v)$ or $w = L_{ccw}(v)$ are drawn solid, while other edges are dashed.

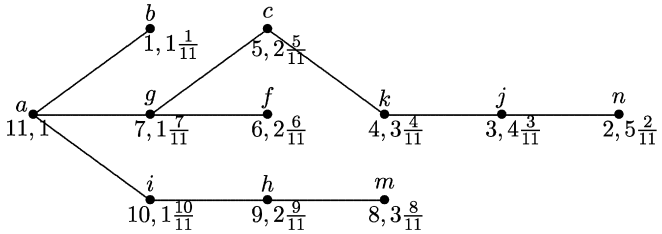


Fig. 4. Breadth first search (from extremal vertex $a$) on the bpg $H$ of Fig. 2. This is simultaneously a breadth first search from leftmost interval $a$ of uig $H^*$. The two numbers below each vertex are its postorder number and its $\phi$ value.

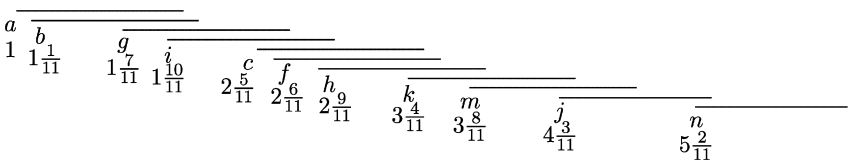


Fig. 5. The uig $H^*$: each unit length interval $x$ has left end at $\phi(x)$.

In [19] a solution for the APSD problem on uig's is described, given a model for the uig. In more detail, given a model for the uig, [19] constructs a BFS and an ordering of vertices, and from these solves the APSD problem by assigning coordinates to intervals, so that distances may be computed by subtracting coordinates (and rounding up to integer). In the next paragraph we briefly describe this solution, starting with a BFS and an ordering of vertices.

A BFS on uig $H^*$ starting from $a$ yields the same partition of vertices and the same BFS structure (i.e., the same interlevel edges) as it does on $H$. As an ordering of the vertices of $H^*$, $\sigma$ satisfies conditions (i) and (ii) in Lemma 3.1 since the BFS structure has not changed from $H$ to $H^*$. The edges selected in [5] as tree edges are the same as here: edges $vw$ such that $w$ is the first vertex (in ordering $\sigma$) that is a neighbor of $v$. In [5] a postorder traversal of $H^*$ was determined, in which for a vertex $w$ with children $v_1$ and $v_2$, if $\sigma(v_1) < \sigma(v_2)$ then $v_1$ was visited before $v_2$; let the postorder traversal number of each vertex $v$ be written postorder($v$). Finally, [5] constructs a model for $H^*$ by setting the coordinate of the left end of interval $v$ (say $v \in L_i$) as $\phi(v) = i + \text{postorder}(v)/n$. Fig. 4 displays the postorder number of each vertex in the graph of Fig. 3, and Fig. 5 displays the model for the corresponding uig $H^*$. In [5] it is shown that for any vertices $x$ and $y$ in $H^*$, $\text{dist}(x, y) = \lceil \phi(x) - \phi(y) \rceil$.

## 5. Algorithm and example

In this section we gather together the parts of the preprocessing algorithm, describe the parallel algorithm, give pseudocode for the query handling algorithm, and give an example.

The first stage of the preprocessing algorithm is a reduction from permutation graph $G$ to a bpg. First it determines the cw-maximal and ccw-maximal vertices of $G$, $V_{cw}$ and $V_{ccw}$. The induced subgraph $H$ of $G$ on $V_{cw} \cup V_{ccw}$ is a bpg. Second, for each vertex $v$ of $G$, it computes $L_{cw}(v)$, $L_{ccw}(v)$, $R_{cw}(v)$, and $R_{ccw}(v)$.

The second step in the preprocessing algorithm is a reduction from bpg $H$ to a uig. To do this, the coordinatization of the uig $H^*$ on vertex set $V_{cw} \cup V_{ccw}$ is computed, as described in Section 4.

Each step of the preprocessing algorithm can be easily parallelized. The preprocessing computations performed in Section 2 use the prefix maxima (and suffix minima) operations; it is well known how to how to compute these using $O(n/\log n)$ processors in $O(\log n)$ time in the EREW PRAM model (where the number of vertices is $n$) [13]. In [19] it is shown how to parallelize the coordinatization of $H^*$ in the same time and processor bounds; the parallel method used is the Euler tour technique [13].

Given vertices $s$ and $t$, the query handling algorithm is to compute $\text{dist}(s, t)$ in $O(1)$ time, and also determine the first internal vertex of a shortest path (if $\text{dist}(s, t) \geqslant 2$).

If $s$ is cw-maximal or ccw-maximal, so $R_{cw}$ or $R_{ccw}$ is nil, in Step 5 only one value for $i$ need be checked. Likewise, if $t$ is cw-maximal or ccw-maximal, then Step 7 simplifies.

Query handling algorithm.

1. Determine if $\text{dist}(s, t) \leqslant 1$; if so, handle directly and return.
2. /* We will presume that $s$ is left of $t$ (otherwise exchange them). */
3. If $R_{cw}(s)$ or $R_{ccw}(s)$ intersects $t$ then return (distance is 2).
4. /* Assert: $\text{dist}(s, t) \geqslant 3$. */
5. For $i \in \{cw, ccw\}$:
6. Let $\alpha$ be the coordinate of $R_i(s)$ in $H^*$.
7. For $j \in \{cw, ccw\}$:
8. Let $\beta$ be the coordinate of $L_j(t)$ in $H^*$.
9. $d_{i,j} = \text{round}(R_i(s), L_j(t), \lceil \beta - \alpha \rceil)$.
10. For the values of $i$ and $j$ giving the minimum $d_{i,j}$:
11. Write ("Distance from $s$ to $t$ is $2 + d_{i,j}$").
12. Write ("First internal vertex in a shortest path is $R_i(s)$.").

For example, given the model for the permutation graph $G$ of Fig. 1, the induced subgraph $H$ of clockwise and counterclockwise maximal segments is displayed in Fig. 2. A model for uig $H^*$ is displayed in Fig. 5. Suppose a query asks for $\text{dist}_G(d, l)$. Segment $d$ intersects segments $c$, $g$, and $i$; $L_{cw}(d) = g$, $R_{cw}(d) = i$, and $L_{ccw}(d) = R_{ccw}(d) = c$. Segment $l$ intersects segments $j$, $m$, and $n$; $L_{cw}(l) = m$, $R_{cw}(l) = n$, and $L_{ccw}(l) = R_{ccw}(l) = j$.

In uig $H^*$, $i$, $c$, $m$, and $j$ are intervals having left end coordinates at $1\frac{10}{11}$, $2\frac{5}{11}$, $3\frac{8}{11}$, and $4\frac{3}{11}$ respectively. The distances in $H^*$ between $R_i(s)$ and $L_j(t)$ for $i, j \in \{cw, ccw\}$ are $\text{dist}_{H^*}(i, m) = \lceil 1\frac{9}{11} \rceil = 2$, $\text{dist}_{H^*}(c, m) = \lceil 1\frac{3}{11} \rceil = 2$, $\text{dist}_{H^*}(i, j) = \lceil 2\frac{4}{11} \rceil = 3$, and $\text{dist}_{H^*}(c, j) = \lceil 1\frac{9}{11} \rceil = 2$. After applying the rounding function, then

$$\text{dist}_G(d, l) = 2 + \min(2 + 0, 2 + 1, 3 + 0, 2 + 0) = 4.$$

So a shortest path from $d$ to $l$ has length 4, and can start and end either $d, i, \ldots, m, l$ or $d, c, \ldots, j, l$.

## References


[1] D. Aingworth, C. Chekuri, P. Indyk, R. Motwani, Fast estimation of diameter and shortest paths (without matrix multiplication), SIAM J. Comput. 28 (1999) 1167–1181.

[2] A. Brandstaedt, V.V. Lozin, On the linear structure and clique-width of bipartite permutation graphs, Ars Combin. 67 (2003) 273–281.

[3] D.Z. Chen, D.T. Lee, R. Sridhar, C.N. Sekharan, Solving the all-pair shortest path query problem on interval and circular arc graphs, Networks 31 (1998) 249–257.

[4] L. Chen, Solving the shortest-paths problem on bipartite permutation graphs efficiently, Inform. Process. Lett. 55 (1995) 259–264.

[5] D.G. Corneil, H. Kim, S. Natarajan, S. Olariu, A.P. Sprague, Simple linear time recognition of unit interval graphs, Inform. Process. Lett. 55 (1995) 99–104.

[6] R.W. Floyd, Algorithm 97: shortest path, Commun. ACM 5 (1962) 345.

[7] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.

[8] K. Han, C.N. Sekharan, R. Sridhar, Unified all-pairs shortest path algorithms, Discrete Appl. Math. 77 (1997) 59–72.

[9] Y. Han, An $O(n^3(\log\log n/\log n)^{5/4})$ time algorithm for all pairs shortest paths, manuscript, 2005.

[10] Y. Han, Improved algorithm for all pairs shortest paths, Inform. Process. Lett. 91 (2004) 245–250.

[11] C.W. Ho, J.-M. Chang, Solving the all-pairs-shortest-length problem on chordal bipartite graphs, Inform. Process. Lett. 69 (1999) 87–93.

[12] F.R. Hsu, M.K. Shan, H.S. Chao, R.C.T. Lee, Some optimal parallel algorithms on interval graphs and circular-arc graphs, J. Inf. Sci Eng. 21 (2005) 627–642.

[13] J. JáJá, An Introduction to Parallel Algorithms, Addison Wesley, Reading, MA, 1992.
[14] P. Mirchandani, A simple $O(n^2)$ algorithm for the all-pairs shortest path problem on an interval graph, Networks 27 (1996) 215–217.
[15] F.P. Preparata, M.I. Shamos, Computational Geometry: an Introduction, Springer, New York, 1988.
[16] R. Ravi, M.V. Marathe, C. Pandu Rangan, An optimal algorithm to solve the all-pair shortest path problem on interval graphs, Networks 22 (1992) 21–35.
[17] F.S. Roberts, Representations of indifference relations, Ph.D. Thesis, Stanford University, 1968.
[18] A.P. Sprague, Recognition of bipartite permutation graphs, Congr. Numer. 62 (1995) 151–161.
[19] A.P. Sprague, T. Takaoka, $O(1)$ time algorithm for all pairs shortest distances on interval graphs, Internat. J. Found. Comput. Sci. 10 (1999) 465–472.